# A Semisupervised Learning Method to Merge Search Engine Results

LUO SI and JAMIE CALLAN
Carnegie Mellon University

The proliferation of searchable text databases on local area networks and the Internet causes the problem of finding information that may be distributed among many disjoint text databases (*distributed information retrieval*). How to merge the results returned by selected databases is an important subproblem of the distributed information retrieval task. Previous research assumed that either resource providers cooperate to provide normalizing statistics or search clients download all retrieved documents and compute normalized scores without cooperation from resource providers.

This article presents a semisupervised learning solution to the result merging problem. The key contribution is the observation that information used to create resource descriptions for resource selection can also be used to create a centralized sample database to guide the normalization of document scores returned by different databases. At retrieval time, the query is sent to the selected databases, which return database-specific document scores, and to a *centralized sample database*, which returns database-independent document scores. Documents that have both a database-specific score and a database-independent score serve as training data for learning to normalize the scores of other documents. An extensive set of experiments demonstrates that this method is more effective than the well-known CORI result-merging algorithm under a variety of conditions.

Categories and Subject Descriptors: H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval—*search process*; H.3.4 [**Information Storage and Retrieval**]: Systems and Software—*distributed systems; information networks*

General Terms: Algorithm, Design, Experimentation

Additional Key Words and Phrases: Distributed information retrieval, semisupervised learning method, resource ranking, resource selection, server selection, results merging

This research was supported by National Science Foundation (NSF) grants EIA-9983253 and IIS-0118767.
Any opinions, findings, conclusions, or recommendations expressed in this article are the authors', and do not necessarily reflect those of the sponsor.
Authors' address: Language Technologies Institute, School of Computer Science, Carnegie Mellon University, 5000 Forbes Avenue, 4502 Newell Simon Hall, Pittsburgh, PA 15213-3890; email: lsi@cs.cmu.edu; callan@cs.cmu.edu.

## 1. INTRODUCTION

The problem of finding information that is scattered among many text databases has become more serious as the number of searchable databases on local area networks and the Internet has increased. The goal is to provide a single, uniform search interface that provides access to all of the searchable text databases available at a given moment. This problem of *distributed information retrieval*, sometimes also called *federated search*, involves building resource descriptions for each database, choosing which databases to search for a particular information need, translating the information need into a form suitable for each selected database, and merging of retrieved results into a single result list that a person can browse easily [Callan 2000].

There are many variants of the general problem, depending upon the degree of cooperation that can be assumed among the service providers. For example, each text database in a small company may run the same software and may cooperate closely to provide an integrated search experience (the *single engine-type* case). In a larger organization, there may be several types of search engine software, but it may be known which databases use which types of software, and there may be limited cooperation among search engines. On the Internet it may not be known which type of software a database uses, and it is unlikely that different service providers will cooperate except in the most rudimentary manner (the *multiple engine-types* case). Each of these environments offers different possibilities and difficulties for a distributed search solution, but all are instances of the more general problem.

In this article, we focus mainly on environments that contain multiple types of independent, uncooperative search engines. This type of environment is found on the Internet and in large organizations. In such an environment one can only assume that individual search engines will run unstructured ("bag of words") text queries and return a list of documents that can be downloaded. One cannot assume that databases will provide any other information about their contents or capabilities.

Distributed information retrieval is commonly viewed as consisting of four subproblems, described below.

—*Resource Description.* How to learn and describe the topics covered by each different database.
—*Resource Selection.* Given an information need (a query) and a set of resource descriptions, how to select a set of resources (text databases) to search.
—*Query Translation.* Given an information need in some base representation, how to map it automatically to representations (query languages and vocabularies) appropriate for the selected databases.
—*Result Merging.* After result lists have been returned from the selected databases, how to integrate them into a single ranked list.

Each of these problems has been addressed to varying degrees in prior research. We survey the prior research in Section 2.

This article presents a solution to the result merging problem. Result merging has received limited attention in prior research, in part because of an assumption that it is similar to the meta-search problem. In meta-search, multiple retrieval algorithms are applied to a single database or to similar databases, producing multiple result lists; the multiple result lists are merged to create a single result list [Aslam and Montague 2001; Lee 1997; Manmatha et al. 2001]. We argue that meta-search and result merging are actually distinct problems.[1] Meta-search solutions depend on the fact that there are multiple scores for a single document, that is, that all of the retrieved lists contain many of the same documents. In contrast, result merging algorithms for distributed IR are applied in environments where there is little or no overlap in the contents of the selected databases, and where it is unusual for two independent search engines to return the same document.

The result-merging problem is difficult because document scores returned by different databases cannot be compared directly. There are two reasons that different databases produce incomparable scores: (i) each database may be searched by a different search algorithm, and (ii) the corpus statistics (e.g., average document length, inverse document frequency, etc.) used to calculate document scores are usually quite different in different databases. If the same document appeared in each database, no two databases would assign it the same score.

Result merging algorithms are usually compared to a single-database baseline; the goal is for a result merging algorithm to produce a single ranked list that is comparable to what would have been produced if the documents from all available databases were combined into a single database. Solutions that depend upon exchanging corpus statistics among available databases and solutions that depend upon downloading documents and calculating new, normalized document scores are essentially trying to estimate the scores that would have been produced by this single-database baseline.

In this article, we discuss an approach to result merging based on semisupervised learning. Our approach adaptively learns to map the database-specific document scores returned from different databases to normalized document scores that approximate single-database baseline document scores. Those scores are directly comparable and can be used to merge the results from different databases into a single ranked list.

This article revisits and extends preliminary research reported in Si and Callan [2002]. In particular, this article includes additional experiments that explore the effectiveness of the semisupervised approach to result merging under conditions that more closely simulate what might be expected in operational or "real world" environments. These experiments are intended to address the criticism that the preliminary efforts required too much transfer of data to be practical in many environments. The results reported here demonstrate that the method is effective even with moderate amounts of transferred data. We also demonstrate that the resource selection and supervision required by the

[1]Some authors define meta-search more broadly, as a unified search interface that queries multiple resources that may or may not overlap. This broader definition subsumes what we call meta-search and some aspects of distributed information retrieval, although it usually assumes that all available resources are searched (i.e., no resource selection).

algorithm can be provided by more than one type of resource selection and ad-hoc retrieval algorithm.

The next section reviews prior research on distributed information retrieval. Section 3 describes the semisupervised result merging algorithm in more detail. Section 4 explains our experimental methodology. Sections 5 and 6 present experimental results for the multiple search engine types and single search engine type cases. Section 7 demonstrates that the method does not depend on using any particular algorithm for resource selection or ad-hoc search. Section 8 investigates how varying the amount of training data affects the accuracy of the merged result list. Section 9 introduces a new version of the algorithm that requires much less training data at the cost of downloading a few documents from selected databases. Section 10 concludes.

## 2. PRIOR RESEARCH

There has been considerable research on distributed information retrieval during the last decade. We survey the prior research in this section, with an emphasis on prior result-merging research and on algorithms that are important to the semisupervised learning solution that we will present later in this paper.

The first problem one faces in developing a distributed IR or federated search system is describing the available resources. STARTS [Gravano et al. 1997] is a protocol for acquiring and describing database contents from cooperative resource providers. It specifies how resource providers should describe the contents of their databases, primarily by providing vocabulary and term frequency information. The STARTS protocol assumes that all providers will identify and count terms in a consistent manner, and not intentionally misrepresent their contents.

When databases are controlled by multiple parties, query-based sampling is a more effective method of acquiring resource descriptions. Resource descriptions are created by sending simple queries to each resource provider and examining the returned documents [Callan and Connell 2001]. Prior research has focused on resource descriptions consisting of vocabulary and term frequency information, but in principle any kind of resource description can be learned. Query-based sampling assumes that databases will run queries and return documents, but no other form of cooperation is assumed. Experiments have shown that under a wide variety of conditions accurate resource descriptions can be learned using about 75 queries to obtain about 300 documents [Callan 2000; Callan and Connell 2001; Craswell et al. 2000].

About a dozen resource selection algorithms have been described in the literature, including GlOSS [Gravano et al. 1999], gGlOSS/vGlOSS [Gravano et al. 1994], CORI [Callan 2000; Callan and Connell 2001], RDD [Voorhees et al. 1995], query clustering [Voorhees et al. 1995], CVV [Craswell et al. 2000], query probing [Hawking and Thistlewaite 1999], and a KL-divergence algorithm [Xu and Croft 1999]. GlOSS, gGlOSS/vGlOSS, CORI, CVV, and the KL-divergence algorithm are based on query-independent resource descriptions that are created in advance of seeing any particular query. Query probing is based on obtaining short, query-specific resource descriptions from each database after a query is received. RDD and query clustering rely on resource descriptions that consist in part of a set of training queries and relevance judgments indicating which databases are the best choices for each training query. Ipeirotis's and Gravano's Hierarchical Database Sampling and Selection algorithms [Ipeirotis and Gravano 2002] try to estimate the frequencies of database words. They use information from the search engine to get document frequencies for some words and then estimate the document frequencies of other words by using the relationship between the rank and the frequency of a word indicated by Mandelbrot's law. The document frequency information is used as a part of the database description to build a hierarchical structure for the databases.

Our research interest is resource selection algorithms that have moderate communications costs, which do not require training data obtained manually, and that operate on full-text queries, which restricts our attention to gGlOSS/vGlOSS, CORI, CVV, and the KL-divergence algorithm. Prior research has shown that the CORI algorithm is effective and stable in a wide variety of environments [Callan 2000; Callan and Connell 2001; Callan et al. 1995b; Craswell et al. 2000; French et al. 1999; Larkey et al. 2000; Powell et al. 2000; Xu and Callan 1998], so we initially restrict our attention further to just that resource selection algorithm. We revisit this decision in Section 7.

The CORI resource selection algorithm operates on resource descriptions that consist of vocabulary, term frequency, and corpus information [Callan 2000; Callan et al. 1995b]. The CORI algorithm is based on a Bayesian Inference Network Model of information retrieval in which each resource is ranked by the belief $P(Q|C_i)$ that query Q is satisfied given that resource $C_i$ is observed (searched). The belief $P(Q|C_i)$ in the simple case relevant to the research reported here is the average of the beliefs $P(r_k|C_i)$ for all query terms $r_k$ in Q. The belief $P(r_k|C_i)$ is estimated as [Callan et al. 1995b]:

$$T = \frac{df}{df + 50 + 150 * cw / avg\_cw} \tag{1}$$

$$I = \frac{\log(|DB| + 0.5)/cf}{\log(|DB| + 1.0)} \tag{2}$$

$$p(r_k|C_i) = b + (1 - b) * T * I, \tag{3}$$

where:

| | |
|---|---|
| df | is the number of documents in $C_i$ that contain $r_k$; |
| cf | is the number of databases that contain $r_k$; |
| \|DB\| | is the number of databases to be ranked; |
| cw | is the number of term occurrences in $C_i$; |
| avg_cw | is the average cw of the databases to be ranked; and |
| b | is the default belief, usually 0.4. |

After a set of databases is ranked with a resource selection algorithm, the most common choice is to select the top N databases or top-scoring databases for search (e.g., Callan [2000], Callan et al. [1995b], Craswell et al. [2000], and Hawking and Thistlewaite [1999]), although more sophisticated decision criteria have also been used (e.g., Fuhr [1999]). In this article, we adopt the common choice of searching the top N databases.

Merging ranked lists returned by different search engines is a difficult task because different retrieval algorithms may be used to search the databases, and because different databases have different corpus statistics. The latter problem has received more attention.

Perhaps the simplest solution, often found in commercial environments, is for each database to use the same retrieval algorithm and the same corpus statistics, for example, by imposing a common set of corpus statistics on all databases. A related solution is for each database to broadcast some or all of its corpus statistics to other databases, so that all can use a common set of global corpus statistics when computing document scores [Xu and Croft 1999]. Another variant on this theme is for each database to return basic term frequency information for each retrieved document and each query term so that the search client can compute a consistent set of document scores using global corpus statistics [Kirsch 2003]. These solutions are all quite effective, but they require significant cooperation, homogeneity, and communication among resource providers.

Another simple result-merging algorithm is to merge results based on the document scores; this is sometimes called the "raw score merge" to emphasize that the database-specific document scores are not normalized in any way. It is most effective when each database uses the same retrieval algorithm, and when corpus statistics are relatively similar among databases. Another simple alternative is *Round Robin*, in which the first document in the merged list is the first document returned by the first selected database, the second document in the merged list is the first document returned by the second selected database, and so on [Voorhees et al. 1995]. Round Robin is a common choice when the database-specific document scores are completely incompatible, for example when different search algorithms are used at different databases, but it has the disadvantage that "weak" databases contribute as many documents to the merged ranking as "strong" databases. Weighted Round Robin, in which databases contribute documents in proportion to their expected value, is a more effective choice [Voorhees et al. 1995].

The result merging algorithm associated with the CORI resource selection algorithm uses a simple heuristic to normalize database-specific document scores. The result merging algorithm[2] is a combination of the score of the database and the database-specific score of the document. The "normalized" score suitable for merging is calculated as shown below [Callan 2000; Callan et al. 1995b]:

$$C_i' = \frac{(C_i - C_{\min})}{(C_{\max} - C_{\min})} \tag{4}$$

$$D' = \frac{(D - D_{\min})}{(D_{\max} - D_{\min})} \tag{5}$$

[2]The algorithm has no specific name in prior publications. In this article, we call it the "CORI result merging" algorithm, because its only prior use is with the CORI resource selection algorithm.

$$D'' = \frac{D' + 0.4 * D' * C_i'}{1.4}. \tag{6}$$

Equation (4) normalizes database scores to the range [0, 1]. If T in Eq. (1) is set to 1.0 for each query term, a score $C_{\max}$ can be computed for each query. If T is set to 0.0 for each query term, a score $C_{\min}$ can be computed for each query. These are the highest and lowest scores that the resource-ranking algorithm could potentially assign to a database. Equation (4) can be calculated from (only) information in the resource selection index. Equation (5) tries to normalize database-specific document scores to the range [0, 1]. It needs the individual search engines to cooperate by providing $D_{\max}$and $D_{\min}$, which are the maximum and minimum scores that could be assigned to any document in that database. In the absence of cooperation, $D_{\max}$ is set to the maximum document score returned by the search engine and $D_{\min}$ is set to the minimum. Equation (6) calculates a database-independent document score suitable for result merging.

Implicit in the CORI result merging algorithm is the assumption that the individual search engines produce database-specific document scores that are relatively similar. In practice, this algorithm is limited to use with the INQUERY search engine [Callan et al. 1995a]. The document score D in Eq. (5) is the database-specific score returned by INQUERY. INQUERY ranks documents by the belief $P(Q|D_i)$ that query Q is satisfied given that document $D_i$ is observed. For simple "bag of words" queries, $P(Q|D_i)$ is determined by the average value of the beliefs of all query items in the corresponding document. The belief $P(r_k|D_i)$ in document $D_i$ according to the query term $r_k$ is determined [Callan et al. 1995a] by:

$$T = \frac{tf}{tf + 0.5 + 1.5 * dw/avg\_dw} \tag{7}$$

$$I = \frac{\log(|D| + 0.5/df)}{\log(|D| + 1.0)} \tag{8}$$

$$p(r_k|D_i) = b + (1 - b) * T * I, \tag{9}$$

where:

| | |
|---|---|
| tf | is the term frequency of word rk in the document; |
| df | is the number of documents that contains the $r_k$; |
| \|D\| | is the number of documents to be ranked; |
| dw | is the number of word occurrences in the document; |
| avg_dw | is the average dw of all the documents; and |
| b | is the default belief, usually 0.4. |

The CORI resource selection method is a variant of the INQUERY retrieval algorithm and the Okapi term-weighting formula [Robertson and Walker 1994]. Equation (1) replaces term frequency in Equation (7) by document frequency, and the constants are scaled by a factor of 100 to accommodate the larger df values. Equation (2) replaces the number of documents in Equation (8) by the number of resources.

The retrieved document score is determined by both the T (tf) and I (idf) parts. For a specific document, the difference between the single-database document score and the database-specific document score lies only in the I (idf) part of the equation. If a database has many documents relevant to a query, then its idf scores tend to be lower than the corresponding centralized idf scores given by Eq. (8). Hence, a document in a highly relevant database tends to have a database-specific score lower than its single-database (or database-independent) score. Equations (5) and (6) try to compensate for the difference by favoring documents in databases that score highly for the query. Therefore, it can be seen that the CORI merging algorithm also tries to mimic the effect of retrieving from a single, global database.

## 3. THE SEMISUPERVISED LEARNING MODEL FOR RESULT MERGING

Most of the result merging solutions described in Section 2 are designed to take place at the search client. However, this makes a difficult problem even more difficult, because result merging is viewed in isolation from the rest of the distributed information retrieval environment. The information available for normalizing database-specific document scores is very limited, and so solutions are heuristic, make strong assumptions, or acquire additional information via additional communication.

Our result merging goal is to efficiently produce a single ranked list of documents that closely approximates the ranked list that would be produced if all of the documents from all of the databases were stored in a single, global database. We call this the *single-database baseline*. In principle, it is possible to produce a (much) better merged list than this baseline, but no method currently achieves even this baseline level of accuracy.

How might a result-merging algorithm approach the accuracy of the single-database baseline? Result-merging algorithms suffer from, and compensate for, the limited information available at the search client. An alternative is to move result-merging from the search client, where information is limited, to the part of the architecture where resource selection occurs. The resource selection component of a distributed IR system has considerable information about the contents of each database. Some of that information might also be useful for merging results.

### 3.1 The Centralized Sample Database

Query-based sampling (Section 2) is a method of creating resource descriptions by sending queries to a database and examining the documents that are returned. Query-based sampling does not assume cooperation among resource providers, so it can be applied in almost any distributed retrieval environment. Experimental results show that it creates resource descriptions that enable accurate resource selection [Callan 2000; Callan and Connell 2001; Craswell et al. 2000].

Usually the documents downloaded during query-based sampling are discarded after the resource descriptions are built, because there is no need to retain them for resource selection. However, these documents can also serve another purpose. The documents obtained by sampling all of the available databases can be combined into a single searchable database. This *centralized sample database* is a representative sample of the single global database that would be created if all of the documents from all of the databases were combined into a single database. The vocabulary and frequency information in the centralized sample database would be expected to approximate the vocabulary and frequency patterns across the complete set of available databases.

## 3.2 Semisupervised Learning

During distributed information retrieval a person enters a query, the query is used to rank the available databases, a set of databases is selected, the query is broadcast to the selected databases, and ranked lists of document ids and scores are returned from each database. These ranked lists of document ids and scores are usually the only input to a result-merging algorithm.

The *semisupervised learning* (SSL) approach to merging results adds an additional step and an additional source of input. When the query is broadcast to the selected databases, it is also sent in parallel to the centralized sample database. The centralized sample database returns a ranked list of document ids and scores. This ranked list of *database-independent* document scores is also provided to the result-merging algorithm.

The database-independent document scores from the centralized sample database are important because they are a good approximation of the scores those documents would have received if they had been retrieved from the (mythical) single global database. The database-independent scores (from the centralized sample database) and the database-specific scores (from the selected databases) can be used to teach a machine learning algorithm how to transform database-specific scores into database-independent scores. We call this method "semisupervised learning" (SSL) because the centralized sample database creates training data that is used by a supervised machine learning algorithm.

The semisupervised learning (SSL) approach to merging results is based on two assumptions: (i) some of the documents retrieved from each selected database will also be retrieved from the centralized sample database; and (ii) given pairs of database-specific and database-independent scores for a small number of documents it is possible to learn functions that accurately map all database-specific scores into their corresponding database-independent scores. The first assumption might seem unlikely to be true. However, recall that resource descriptions are created from documents that ranked highly for some query during query-based sampling of a database. Those documents then become part of the centralized sample database. A database is only selected for search when its resource description, created from sampled documents that are also in the centralized sample database, closely matches the query. We show in Section 5.1 that these "overlap" documents are more common than one might expect.

Given pairs of database-specific and database-independent scores for a set of "overlap" documents, the problem becomes how to learn a function that maps all database-specific scores into their corresponding database-independent scores.

Regression is an efficient and effective mathematical tool for this kind of problem. First, we assume the type of the mapping function, and then the parameters in the mapping function can be determined by a minimum squared error criterion. It can be formally described as:

$$\lambda = \arg\max_{\lambda} \sum_{i} (f(\lambda, x_i) - y_i)^2, \tag{10}$$

where $\lambda$ is the set of parameters, $f(\lambda, x_i)$ is some type of function, and $(x_i, y_i)$ is the $i$th training data point, which in this case is the $i$th pair of (database-specific, database-independent) document scores.

It is an open question what type of mapping function would be best. Linear mapping functions have two favorable characteristics: (i) the CORI result merging algorithm is a linear function, hence there is evidence that a linear model can be effective; and (ii) linear functions can be computed efficiently from small amounts of training data because there are few (only two) parameters. The work reported in this paper is based on learning linear models.

The SSL approach to merging results depends upon the presence of training data, so it is necessary to specify what happens when there is too little training data (i.e., when there are not enough "overlap" documents). When there is insufficient training data, the SSL approach "backs off" to the heuristic CORI result merging algorithm (Section 2). This solution is not ideal, but there is considerable empirical evidence from prior research showing that it is reasonably effective. We revisit this issue in Sections 5.1 and 9.

After the normalizing models are built, the database-specific document scores are transformed to their corresponding database-independent document scores. The rankings are merged into a single final list by comparing database-independent document scores.

When databases are searched by multiple (or unknown) search engines (the *multiple engine-types* case), it is unlikely that a single function can transform from any database-specific score to a database-independent score. In these cases it may be more effective to build different normalizing models for each database. When it is known that all databases are searched by the same type of search engine (the *single engine-type* case), it may be possible to combine all the training data together to build a single normalizing model. Our main research focus is the multiple engine-types case, so we describe it first.

## 3.3 The Multiple Search Engine-Types Case

In the multiple search engine-types case, different normalizing models should be built for different databases. The first step is to identify documents that appeared in the result list returned by the centralized sample database *and* in the result list of one of the selected databases; we call these *overlap* documents. For each overlap document $d_{i,j}$ from database $C_i$, the pairs of database-specific and centralized database document scores $S_i(d_{i,j})$ and $S_C(d_{i,j})$ are the training data for the linear model. The goal is to find a linear function $S_C(d_{i,j}) = a_i * S_i(d_{i,j}) + b_i$ that maps database-specific document scores to centralized database document scores (which approximate database-independent scores). The regression over all training data from a selected database $C_i$ can be shown in matrix representation as:

$$\begin{bmatrix} S_i(d_{i,1}) & 1 \\ S_i(d_{i,2}) & 1 \\ \cdots & 1 \\ S_i(d_{i,n}) & 1 \end{bmatrix} * [a_i \ \ b_i] = \begin{bmatrix} S_C(d_{i,1}) \\ S_C(d_{i,2}) \\ \cdots \\ S_C(d_{i,n}) \end{bmatrix}, \tag{11}$$

where $a_i$ and $b_i$ are the parameters in the linear transformation. Call these matrices $X$ (the first item on the left side of Eq. (11), which is constructed from database-specific scores and constants), $W$ (the second item on the left side of Eq. (11), which is the set of parameters) and $Y$ (the item on the right side of the equation, which is the set of centralized sample database scores). Simple algebraic manipulation allows the solution to be expressed as shown below.

$$W = (X^T X)^{-1}(Y^T X). \tag{12}$$

Equation (12) is the solution under the Minimum Squared Error (MSE) criterion. Each model maps all of the document scores from a particular database to the database-independent scores approximated by the centralized sample database. The database-independent document scores calculated by the models for different databases are comparable, and so can be used to generate the final merged ranked list.

3.3.1 *Model Training and Adjustment.* There are two parameters in a single linear model. At least two data points are needed to train a linear model. More training data usually generates more accurate models, so we require at least three training points to fit a linear model. When there are fewer than three training points available for a database (i.e., fewer than three "overlap" documents), we call this database a "bad" database.

Our hypothesis is that a database that contains a lot of relevant documents will tend to also have a lot of overlap documents. When there are many overlap documents for a database, it is possible to be selective about which are used for training. We believe that it is more important to be more precise at the top of the merged ranking, so only the top 10 overlap documents (as determined by the selected database) are used to train the model. If a database has less than 10 overlap documents, all of them are used.

If too many of the selected databases don't return enough overlap documents for a specific query, it may indicate that the semisupervised learning approach to result merging is a poor choice for the query. This might happen, for example, in a situation where the query is not a particularly good match for any of the available databases. We arbitrarily set 40% as a threshold; if the percentage of databases with fewer than three overlap documents exceeds 40% of the selected databases, the CORI result merging algorithm (Eqs. (4)–(6)) is used to merge results instead of the semisupervised learning method. In Section 5 we show that this pathological condition is rare. In Section 9, we present an alternative solution for generating training data when it does arise, making the back-off strategy unnecessary.

It is possible for the regression algorithm to create linear models that are anomalous for ad-hoc document retrieval. A model is anomalous if the learned linear model produces a database-independent document score greater than 1 for some retrieved document. A score above 1 is not a serious problem by itself (i.e., the score could still be used for merging result lists), but it signals that the model is biased high and will give the documents returned from that database too much of an advantage.

This problem is addressed by replacing the original linear model with the closest new model that intersects (1,1). Let $y = ax + b$ be the original model, and $y' = a'x + b'$ be the adjusted model, then the problem can be expressed as shown below.

$$(a', b') = \underset{a', b'}{\arg\min} \int_0^1 [(a' - a) * x + (b' - b)]^2 \, dx. \tag{13}$$

Simple mathematical manipulation yields the following expression.

$$a' = \frac{3 - a - 3b}{2} \qquad b' = 1 - a'. \tag{14}$$

This bias correction produces slightly more accurate results in our experiments.

## 3.4 The Single Search Engine Type Case

In some environments, (e.g., a local area network or an Intranet), it is known that each text database is searched by the same type of search engine. The CORI result-merging algorithm is evidence that in at least some of these "single engine-type" cases a single model can be used to merge results returned from different databases. Our hypothesis is that using all of the training data to train a single model for the single search engine type case will produce more accurate results than dividing the training data to create multiple, database-specific models. This hypothesis is tested in Section 6.

The CORI merging algorithm (Eq. (6)) normalizes database-specific document scores by a linear combination of the database-specific document score and the database weight. For consistency, we re-express it in the same notation used to express the multiple search engine types version of the SSL merging algorithm (Eq. (11)).

$$S'_C(d_{i,j}) = \frac{S_i(d_{i,j}) + 0.4 \times S_i(d_{i,j}) \times S(C_i)}{1.4}. \tag{15}$$

In Eq. (15), S(C$_i$) is the database score for $i$th database (Eq. (4) and $S_i(d_{i,j})$ is the database-specific score for the $j$th document retrieved from that database (Eq. (5)). In the CORI result merging algorithm, one constant parameter controls the influence of the database score. Our approach is based on the hypothesis that the merging algorithm will be more accurate if this parameter is tuned on a query-by-query basis.

As in the multiple search engine types case, the first training step is to identify documents that appeared in the result list returned by the centralized sample database *and* in the result list of one of the selected databases ("overlap" documents). For each overlap document $d_{i,j}$ from any selected database $C_i$, the pairs of database-specific and centralized database document scores $S_i(d_{i,j})$ and $S_C(d_{i,j})$ are the training data for the linear model. All of the training data is used to train a single linear model that is used to normalize scores from all databases. The regression over all training data from the set of selected databases can be expressed in matrix representation as shown below.

$$\begin{bmatrix} S_1(d_{1,1}) & S(C_1)S_1(d_{1,1}) \\ S_1(d_{1,2}) & S(C_1)S_1(d_{1,2}) \\ \cdots\cdots & \cdots\cdots\cdots\cdots \\ S_n(d_{n,m}) & S(C_n)S_n(d_{n,m}) \end{bmatrix} * [a\ b] = \begin{bmatrix} S_C(d_{1,1}) \\ S_C(d_{1,2}) \\ \cdots\cdots \\ S_C(d_{n,m}) \end{bmatrix}. \tag{16}$$

Call these matrices $X$, $W$ and $Y$ as in Eq. (11). The solution (Eq. (12)) is a single linear function that maps database-specific document scores from all databases to database-independent scores that approximate estimated centralized document scores as shown below:

$$S'_C(d_{i,j}) = a * S_i(d_{i,j}) + b \times S_i(d_{i,j}) \times S(C_i). \tag{17}$$

The database-independent document scores calculated by Eq. (17) are comparable, and so can be used to generate the final merged ranked list.

3.4.1 *Model Training and Adjustment.* As in the multiple search engine-types case, only the top retrieved documents are used to build the regression model. Each database is allowed to contribute a maximum of 20 "overlap" documents to the training pool; this limit was determined empirically. If there is insufficient data (fewer than three "overlap" documents) to train a linear model, the algorithm backs off to the CORI result-merging algorithm. Backing off would be expected to occur only very rarely (e.g., it never occurred in our experiments) because all of the training data from all of the selected databases is used to train the model.

## 4. EXPERIMENTAL METHODOLOGY

The SSL approach to result merging was tested with a variety of experiments using two testbeds and several search engine algorithms. This section describes the experimental methodology. The next sections describe the experiments.

### 4.1 Testbeds

Testbeds (documents, queries, relevance judgments) play an important role in distributed information retrieval experiments, because the performance of a distributed information retrieval system is highly influenced by the testbed characteristics. Two types of testbeds were used in our experiments (Tables I and II).

(1) *Organized by Source and Date* (*trec123*). One hundred databases were created from TREC CDs 1, 2 and 3. This testbed models an approach to managing databases that is common in some operational environments because it is easy to administer. Documents are assigned to databases based on source and publication date [Callan 2000]. The databases are somewhat

Table I. Testbed Statistics

| Testbed Name | Testbed Size (GB) | Database Sizes | | | | | |
|---|---|---|---|---|---|---|---|
| | | Number of Documents | | | Megabytes (MB) | | |
| | | Min | Avg | Max | Min | Avg | Max |
| Trec123 | 3.2 | 752 | 10,782 | 39,713 | 28.1 | 32 | 41.8 |
| Trec4_kmeans | 2.0 | 301 | 5,675 | 82,727 | 3.9 | 20 | 248.6 |

Table II. Query Set Statistics

| Testbed Name | TREC Topic Set | TREC Topic Field | Average Length (Words) |
|---|---|---|---|
| Trec123 | 51–100 | Title | 3.0 |
| Trec4 | 201–250 | Description | 7.2 |

heterogeneous. Fifty short queries were extracted from the title fields of TREC topics 51–100 automatically.

(2) *Organized by Topic* (*trec4_kmeans*). One hundred databases were created from TREC4 ad-hoc data. This testbed models an environment in which databases are organized by topic and are relatively homogeneous. A $k$-means clustering algorithm was used to cluster the databases by topic automatically [Xu and Croft 1999]. Databases are homogenous and the word distributions are very skewed. 50 longer queries were extracted from the description fields of TREC topics 201–250 automatically.

Summary statistics for the two testbeds and their queries are shown in Tables I and II.

The two different testbeds provide different advantages and disadvantages to the different components of a distributed IR system. The trec4_kmeans testbed is generally considered easier for resource selection because the queries are longer and because the databases were constructed to have homogeneous content and to be relatively different from each other [Xu and Croft 1999]. The databases are easier to tell apart, and because databases are relatively homogeneous, the top $N$ databases tend to cover a higher percentage of relevant documents than the top $N$ trec123 databases. The trec123 testbed is generally considered easier for result merging, because databases are more similar to each other; a document ranked $n$th by one database is often of similar quality to a document ranked $n$th by a similar database. However, as result-merging algorithms become more sophisticated, the relative result-merging difficulty of the two testbeds becomes less clear. The trec4_kmeans databases tend to be smaller than the trec123 databases; hence a query-based sampling procedure that draws a fixed number of documents from each database covers a larger fraction of the trec4_kmeans databases. The queries used with the trec4_kmeans testbed are also longer than the queries used with the trec123 testbed. These differences might provide more or better quality training data for the SSL result-merging algorithm on the trec4_kmeans testbed.

## 4.2 Search Engines

In the multiple search engine types environment different search engines may be used for each resource. Three different kinds of search engines were used in our experiments: INQUERY [Callan et al. 1995a], a statistical language modeling algorithm [Lemur Toolkit 2003; Ogilvie and Callan 2001], and a Vector–Space algorithm similar to SMART [Buckley et al. 1995]. All three are well known and effective retrieval algorithms. The retrieval algorithm used in INQUERY was described in Section 2. The Vector—Space search engine used SMART “ltc” weighting [Buckley et al. 1995]. The Language Model retrieval algorithm is relatively new, so it is described briefly below.

In the Language Model retrieval algorithm, each document is considered as a sample of text generated from a specific language. The language model for each document is estimated based on the word frequency patterns observed in the text. The relevance of a document to a query is estimated by how likely the query is to be generated by the language model that generated the document. More specifically, the likelihood of a query $Q$ to be generated from the language model of document $D$ is computed as:

$$P(Q|D) = \prod_{q \in Q} (\lambda P(q|D) + (1 - \lambda) P(q|C)), \tag{18}$$

where $q$ is a query term in the query $Q$, $P(q|D)$ is the multinomial distribution of the document language model, $P(q|C)$ is the multinomial distribution of the database $C$ to which the document $D$ belongs, and $\lambda$ is a weighting parameter between 0 and 1. In all our experiments, $\lambda$ is set to 0.5.

Although the three algorithms are somewhat similar in their effectiveness, they produce relatively different document scores. Scores from INQUERY can vary in the range [0.4, 1.0], but typically fall within the range [0.4, 0.7]. Document scores from the Language Model retrieval method are negative numbers, usually in the range [−60, −30]. SMART document scores can range over [0.0, 1.0], but typically fall within [0.0, 0.3].

## 4.3 Distributed Retrieval Environment

In our experiments, all resource descriptions were created using query-based sampling. All sampling queries were one word long. The initial sampling query was selected randomly from a list of English words. Subsequent sampling queries were selected randomly from the resource description being learned for the database (i.e., a term seen in a document retrieved earlier from that database). The top four documents returned for each query were used to create the resource description. Sampling continued until 300 unique documents were seen for the database (usually about 75 queries). This methodology is consistent with our prior research on query-based sampling [Callan 2000; Callan and Connell 2001]. Experiments in Section 8 examine the effect of changing the number of documents sampled.

The documents obtained from different databases were also used to form a centralized sample database of 30,000 documents (100 databases $\times$ 300 documents per database). The INQUERY retrieval algorithm was the default search algorithm for the centralized sample database. Experiments described in Section 7 examine the effect of changing the algorithm used to search the centralized sample database.

The CORI resource-ranking algorithm was used to rank the databases for each query. Section 7 examines the effect of changing the resource selection algorithm. The top 10 databases were selected for search, and were searched by the search engines described above (Section 4.2). Each database returned a result list consisting of document identifiers and scores for the top-ranked 1,000 documents, unless otherwise specified. Experiments in Section 9 examine the effect of varying the lengths of result lists.

Effectiveness in these experiments was measured by Precision at specified ranks between 5 and 30. This choice was influenced by our interest in interactive retrieval environments, where users rarely search below 30 documents. It was also influenced by the characteristics of the distributed retrieval task. When most databases are not searched, metrics such as Average Precision, which measure effectiveness across a wide range of document ranks, tend to be affected strongly by testbed characteristics and the percentage of databases searched. Testbeds that have relevant documents clustered in a few databases will have much higher Average Precision than testbeds where relevant documents are scattered across more databases. Precision at low Recall tends to be less affected by these characteristics.

## 5. EXPERIMENTAL RESULTS: MULTIPLE SEARCH ENGINES

A series of experiments was conducted to evaluate the performance of the semisupervised learning (SSL) approach to result merging under a variety of conditions. In all of the multiple search engine-types experiments described in this section the databases were distributed evenly among the different types of search engines using "round robin" distribution.

### 5.1 Overlap Documents

A sufficient amount of training data is crucial to the SSL result merging algorithm. The number of overlap documents depends on factors such as the number of documents retrieved from each database during query-based sampling; query characteristics; the characteristics of selected databases; and the number of documents retrieved from each selected database. We expect that the number of overlap documents will be larger when more documents are retrieved during query-based sampling; the number may also be larger when queries are longer, when databases are homogenous, when databases are small, or when more documents are retrieved from each selected database.

An experiment investigated the hypothesis that there are usually sufficient training data for the semi-supervised learning approach to result merging. In this experiment databases were assigned to three types of search engines (INQUERY, Language Model, Vector Space) using round robin assignment. Result lists containing ids and scores of up to 1,000 documents were returned by each database selected for the query, and a result list of up to 1,000 documents was returned by the centralized sample database. The number of overlap documents for each of the queries was counted. Figures 1 and 2 show the number of overlap documents for the queries on the two testbeds.

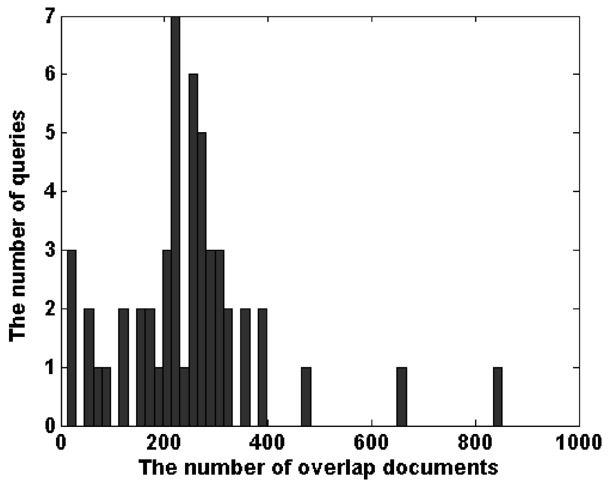

Fig. 1. The distribution of overlap documents for 50 "Title" queries on the trec123 testbed.

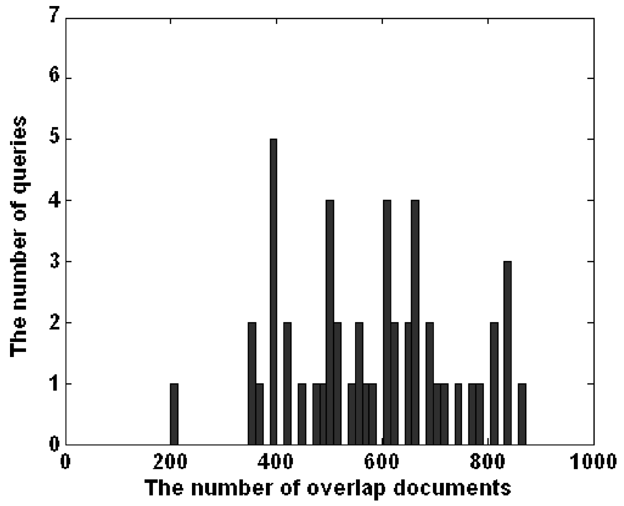

Fig. 2. The distribution of overlap documents for 50 "Description" queries on the trec4_kmeans testbed.

Most of queries (47 out 50) on the trec123 testbed had more than 50 overlap documents. All of the queries on the trec4_kmeans testbed had more than 200 overlap documents. These results were consistent with our expectations. Overlap documents were plentiful for most queries on both testbeds. One would expect the trec4_kmeans testbed to produce more overlap documents per query because its databases are more homogeneous, the average database size is smaller, and the queries were longer and of higher quality.

As described in Section 3.3.1, the result-merging algorithm backs off to the CORI algorithm when there is not enough training data. The algorithm backed

Table III. Precision at Different Document Ranks using the CORI and Semisupervised Learning Approaches to Merging Retrieval Results. INQUERY and Language Model Search Engines

| Document Rank | Trec123 Testbed | | Trec4_kmeans Testbed | |
|---|---|---|---|---|
| | CORI Merge | SSL Merge | CORI Merge | SSL Merge |
| 5 | 0.3600 | 0.4080 (+13.3%) | 0.2640 | 0.4040 (+53.0%) |
| 10 | 0.3460 | 0.3820 (+10.4%) | 0.1900 | 0.3780 (+98.9%) |
| 15 | 0.3480 | 0.3560 (+2.3%) | 0.1840 | 0.3400 (+84.8%) |
| 20 | 0.3400 | 0.3440 (+1.2%) | 0.1800 | 0.3130 (+73.9%) |
| 30 | 0.3247 | 0.3200 (−1.4%) | 0.1647 | 0.2740 (+66.4%) |

*Note*: Ten databases were selected to search for each query. Results are averaged over 50 queries.

Table IV. Precision at Different Document Ranks using the CORI and Semisupervised Learning Approaches to Merging Retrieval Results. INQUERY and Vector—Space Search Engines

| Document Rank | Trec123 Testbed | | Trec4_kmeans Testbed | |
|---|---|---|---|---|
| | CORI Merge | SSL Merge | CORI Merge | SSL Merge |
| 5 | 0.2840 | 0.3280 (+15.5%) | 0.2280 | 0.3360 (+47.3%) |
| 10 | 0.2700 | 0.3040 (+12.6%) | 0.1660 | 0.3080 (+85.5%) |
| 15 | 0.2640 | 0.2947 (+11.6%) | 0.1613 | 0.2760 (+71.1%) |
| 20 | 0.2610 | 0.2820 (+8.0%) | 0.1610 | 0.2620 (+62.7%) |
| 30 | 0.2487 | 0.2867 (+15.3%) | 0.1487 | 0.2240 (+50.6%) |

*Note*: Ten databases were selected to search for each query. Results are averaged over 50 queries.

off to CORI result-merging for 3 out of 50 queries on the trec123 testbed. It didn't back off for any of the queries on the trec4_kmeans testbed. These results suggest that generally there are enough training data for the learning algorithm. We revisit this topic in Section 9.

## 5.2 Comparison with Cori Result-Merging

In order to test the effectiveness of the SSL result-merging algorithm, a set of experiments was conducted with different combinations of search engine types. Experiments were conducted with a combination of three types of search engine, and with three combinations of two types of search engine. Databases were assigned to types of search engines using round robin assignment. The CORI result-merging algorithm (Section 2) was used as a baseline for comparison.

Results for the three "two search engines" experiments are summarized in Tables III–V. On the trec123 testbed, the SSL algorithm was about as good as or better than the CORI result-merging algorithm. The largest improvement, about 10%, was on the combination of INQUERY and the Vector—Space search engine. Results are more dramatic on the trec4_kmeans testbed because it is organized by topic and has very skewed word distributions. Prior research identified this as a difficult testbed for the CORI result-merging algorithm [Larkey et al. 2000], which is confirmed by our experiments. The SSL merge algorithm was far more effective with all combinations of two search engines on this testbed.

Experimental results for the "three engines" experiment, summarized in Table VI, were consistent with the results for "two engines" cases. The SSL

Table V. Precision at Different Document Ranks using the CORI and Semisupervised Learning Approaches to Merging Retrieval results. Language Model and Vector—Space Search Engines

| Document Rank | Trec123 Testbed | | Trec4_kmeans Testbed | |
|---|---|---|---|---|
| | CORI Merge | SSL Merge | CORI Merge | SSL Merge |
| 5 | 0.2880 | 0.2960 (+2.8%) | 0.2120 | 0.3400 (+60.4%) |
| 10 | 0.2680 | 0.2860 (+6.7%) | 0.1800 | 0.3120 (+73.3%) |
| 15 | 0.2693 | 0.2840 (+5.5%) | 0.1680 | 0.2720 (+61.9%) |
| 20 | 0.2680 | 0.2750 (+2.6%) | 0.1620 | 0.2530 (+56.2%) |
| 30 | 0.2620 | 0.2687 (+1.6%) | 0.1507 | 0.2333 (+51.4%) |

*Note*: Ten databases were selected to search for each query. Results are averaged over 50 queries.

Table VI. Precision at Different Document Ranks using the CORI and Semisupervised Learning Approaches to Merging Retrieval Results. INQUERY, Language Model, and Vector–Space Search Engines

| Document Rank | Trec123 Testbed | | Trec4_kmeans Testbed | |
|---|---|---|---|---|
| | CORI Merge | SSL Merge | CORI Merge | SSL Merge |
| 5 | 0.3240 | 0.3520 (+8.6%) | 0.2560 | 0.3640 (+42.2%) |
| 10 | 0.3020 | 0.3400 (+12.6%) | 0.1920 | 0.2940 (+54.4%) |
| 15 | 0.3013 | 0.3280 (+8.9%) | 0.1787 | 0.2760 (+43.5%) |
| 20 | 0.2960 | 0.3290 (+11.1%) | 0.1770 | 0.2540 (+47.4%) |
| 30 | 0.2947 | 0.3200 (+7.0%) | 0.1560 | 0.2300 (+46.0%) |

*Note*: Ten databases were selected to search for each query. Results are averaged over 50 queries.

merge algorithm was a little better than the CORI merging algorithm on the trec123 testbed and was much more effective on the trec4_kmeans testbed.

The experiments reported in Tables III–VI investigated the accuracy of the SSL and CORI merging algorithms when 10 databases were selected for search. Another experiment evaluated the performance of these two algorithms with fewer selected databases. Table VII shows the results when 5 databases were selected in the "three search engines" case.

The SSL algorithm was about as effective as the CORI merging algorithm on the trec123 testbed, but it was much more effective than the CORI merge algorithm on the trec4_kmeans testbed. It is perhaps more interesting to investigate how the two algorithms were affected when the number of databases was reduced. The accuracy of the CORI merging algorithm improved slightly, but the accuracy of the SSL algorithm deteriorated slightly. The improvement of the CORI merging algorithm suggests that it is distracted by lower-quality databases; when they are removed, the result merging task is a little easier, so the results for the CORI merging algorithm improve. The drop in the accuracy of the SSL algorithm suggests that it is relatively good at identifying the few relevant documents in the lower-quality databases; when these databases are removed, the SSL algorithm has fewer relevant documents to work with, so its Precision drops a little.

The learning component of the SSL algorithm dynamically adjusts parameters in a model that is otherwise identical to the model used by the CORI merging algorithm. The CORI merging algorithm uses a constant to control the effect of the database score on the normalized document score; the SSL merging algorithm varies this parameter dynamically, on a database-specific

Table VII. Precision at Different Document Ranks using the CORI and Semisupervised Learning Approaches to Merging Retrieval Results. INQUERY, Language Model, and Vector—Space Search Engines

| Document Rank | Trec123 Testbed | | Trec4_kmeans Testbed | |
|---|---|---|---|---|
| | CORI Merge | SSL Merge | CORI Merge | SSL Merge |
| 5 | 0.3240 | 0.3400 (+4.9%) | 0.2560 | 0.3080 (+20.3%) |
| 10 | 0.3220 | 0.3240 (+0.6%) | 0.2200 | 0.2660 (+19.8%) |
| 15 | 0.3107 | 0.3067 (−1.3%) | 0.2080 | 0.2507 (+23.6%) |
| 20 | 0.3010 | 0.3000 (−0.3%) | 0.1830 | 0.2440 (+33.3%) |
| 30 | 0.2787 | 0.2793 (+0.2%) | 0.1687 | 0.2167 (+28.5%) |

*Note*: Five databases were selected to search for each query. Results are averaged over 50 queries.

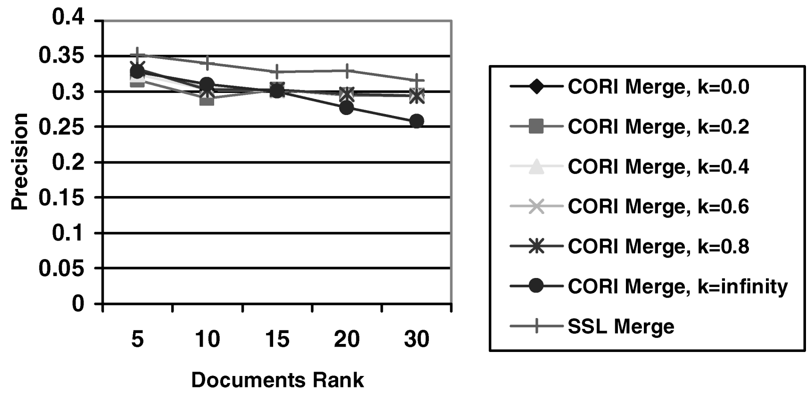

Fig. 3. How varying the k parameter in the CORI result merging algorithm affects Precision on the trec123 testbed.

and query-specific basis. Our hypothesis is that the ability to adjust this parameter dynamically is why the SSL merging algorithm is more effective than the CORI merging algorithm, but the experiments reported above don't test this hypothesis directly. The CORI result-merging algorithm is generally considered well-tuned, but it is possible that some other constant database weighting factor would give better accuracy.

The CORI result-merging algorithm (Equation 6) can be rewritten to make this parameter explicit, as shown below:

$$D'' = \frac{D' \times (1 + k \times C_i')}{1 + k}, \tag{19}$$

$k$ represents the importance of the database score for computing a normalized document score. $k$ is usually set to 0.4 for the CORI result-merging algorithm (Eq. (6)). An experiment was conducted to study the effect of setting $k$ to other values. When $k$ is set to infinity, the formula is $D'' = D'C'$.

Figures 3 and 4 show the effect of using different values of k in the CORI result-merging algorithm. The particular choice of $k$ did not make much difference to the effectiveness of the CORI merging algorithm; all of the choices were about equally good, on average. This experiment confirms the hypothesis that the power of the SSL algorithm is its ability to *dynamically* set k to

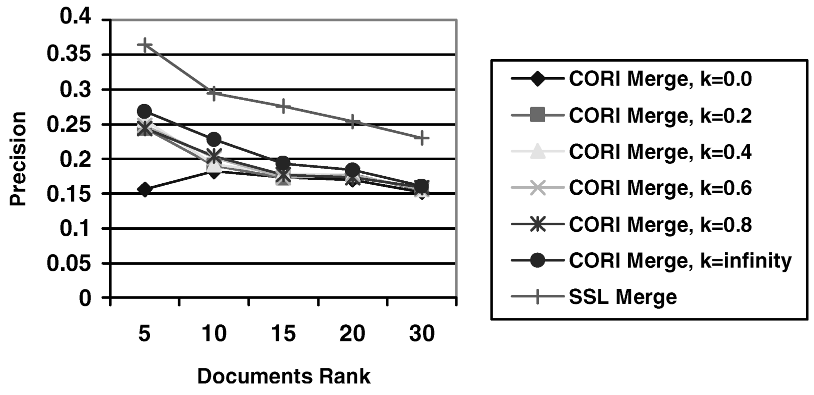

Fig. 4. How varying the k parameter in the CORI result merging algorithm affects Precision on the trec4_kmeans testbed.

Table VIII. Precision at Different Document Ranks using the CORI and Semisupervised Learning Approaches to Merging Retrieval Results from INQUERY, Language Model, and Vector—Space Search Engines that Don't Return Document scores

| Document Rank | Trec123 Testbed | | Trec4_kmeans Testbed | |
|---|---|---|---|---|
| | CORI Merge | SSL Merge | CORI Merge | SSL Merge |
| 5 | 0.2960 | 0.3520 (+18.9%) | 0.2160 | 0.3240 (+50.0%) |
| 10 | 0.2680 | 0.3040 (+13.4%) | 0.1960 | 0.2660 (+35.7%) |
| 15 | 0.2693 | 0.2813 (+4.5%) | 0.1907 | 0.2520 (+32.1%) |
| 20 | 0.2630 | 0.2630 (+0.0%) | 0.1780 | 0.2310 (+29.8%) |
| 30 | 0.2540 | 0.2513 (−1.5%) | 0.1613 | 0.2053 (+27.3%) |

*Note*: Ten databases were selected for each query. Results are averaged over 50 queries.

different values for different queries and databases, instead of having a constant value for all queries and databases.

Another set of experiments was designed to address the scenario where the databases return no document scores but report only ranked lists of results. The CORI and semi-supervised learning approaches were used to merge retrieval results from INQUERY, Language Model, and Vector-Space search engines. Ten databases were selected to search for each query and the results were averaged over 50 queries. However, in these experiments the document scores were eliminated intentionally and the returned result from each database was only a ranked list of document identifiers. Each document was assigned a pseudoscore.[3] These pseudoscores were the input to the CORI and SSL algorithms. Experimental results are shown in Table VIII.

The experiments demonstrate the effectiveness of the SSL algorithm when databases return only ranked lists without document scores. The results show that the SSL algorithm is better than the CORI algorithm in most cases and the improvement is larger on the trec4_kmeans testbed, which is consistent with our prior experiments.

[3]First document has a score of 1; the second has a score of 0.999 etc. At most, 1000 documents were retrieved.

Table IX. Precision at Different Document Ranks using the CORI and Semisupervised Learning Approaches to Merging Retrieval Results. INQUERY Search Engine

| Document Rank | Trec123 Testbed | | Trec4_kmeans Testbed | |
|---|---|---|---|---|
| | CORI Merge | SSL Merge | CORI Merge | SSL Merge |
| 5 | 0.4480 | 0.4680 (+4.5%) | 0.4240 | 0.4520 (+6.6%) |
| 10 | 0.4220 | 0.4360 (+3.3%) | 0.3860 | 0.4060 (+5.2%) |
| 15 | 0.4053 | 0.4107 (+1.3%) | 0.3400 | 0.3573 (+4.6%) |
| 20 | 0.3820 | 0.3840 (+0.5%) | 0.3140 | 0.3230 (+2.9%) |
| 30 | 0.3627 | 0.3647 (+0.6%) | 0.2753 | 0.2913 (+5.8%) |

*Note*: Ten databases were selected to search for each query. Results are averaged over 50 queries.

The experiments reported in this section demonstrate that the SSL algorithm for merging results is an improvement over the CORI result-merging algorithm in environments containing multiple search engines. The SSL algorithm was as effective as the CORI merge algorithm on the "easy" trec123 testbed, and much more effective on the "harder" trec4_kmeans testbed. A comparison of results from searching the top 5 databases and the top 10 databases suggests that the SSL algorithm is better able to merge relevant documents from lower-quality databases. An experiment varying the database weighting parameter $k$ demonstrates that the power of the SSL algorithm is its ability to vary $k$ dynamically on a query-specific and database-specific basis. The SSL algorithm was also shown to be effective when search engines return only document ranks, but not scores.

## 6. EXPERIMENTAL RESULTS: SINGLE SEARCH ENGINE

In a local area network or on an Intranet, it is not unusual for all text databases to be searched by the same type of search engine. For example, the CORI result-merging algorithm assumes that all the search engines are INQUERY, and its performance has been the state-of-the-art in this research area. The set of experiments reported in this section investigate the effectiveness of the single engine-type variant of the semisupervised learning approach to result merging. The main difference between the multiengine and single-engine versions of the algorithm are that the single-engine type combines all of the training data to train a single linear model that transforms database-specific document scores into normalized database-independent document scores.

The first single search engine type experiment compared the single engine-type version of the SSL result-merging algorithm (Section 3.4) to the CORI result-merging algorithm. The INQUERY search engine was used to search selected databases because the CORI result-merging algorithm is tuned for INQUERY. Table IX summarizes the experimental results.

The SSL merging algorithm was about as effective as the CORI merging algorithm in the tests with both testbeds. This experiment demonstrates that the CORI result-merging algorithm is indeed well tuned for environments containing only INQUERY search engines. However, the SSL algorithm compared favorably in what might be considered a best case scenario for the CORI result-merging algorithm.

Table X. Precision at Different Document Ranks using the CORI and Semisupervised Learning Approaches to Merging Retrieval Results. INQUERY Search Engine

| | Trec123 Testbed | | | Trec4_kmeans Testbed | | |
|---|---|---|---|---|---|---|
| Document Rank | CORI Merge | SSL Merge, Single Model | SSL Merge, Multiple Models | CORI Merge | SSL Merge, Single Model | SSL Merge, Multiple Models |
| 5 | 0.4480 | 0.4680 (+4.5%) | 0.3960 (−11.6%) | 0.4240 | 0.4520 (+6.6%) | 0.4760 (+12.26%) |
| 10 | 0.4220 | 0.4360 (+3.3%) | 0.3660 (−13.27%) | 0.3860 | 0.4060 (+5.2%) | 0.3800 (−1.55%) |
| 15 | 0.4053 | 0.4107 (+1.3%) | 0.3600 (−11.18%) | 0.3400 | 0.3573 (+4.6%) | 0.3453 (+1.56%) |
| 20 | 0.3820 | 0.3840 (+0.5%) | 0.3430 (−10.21%) | 0.3140 | 0.3230 (+2.9%) | 0.3310 (+5.41%) |
| 30 | 0.3627 | 0.3647 (+0.6%) | 0.3307 (−8.82%) | 0.2753 | 0.2913 (+5.8%) | 0.2907 (+5.59%) |

*Note*: Ten databases were selected to search for each query. Results are averaged over 50 queries.

The single engine-type variant of the SSL algorithm is based on the hypothesis that it is more effective to combine training data ("overlap" documents) from different databases running the same software. In this case, only a single linear model is learned that must normalize document scores returned by all of the databases. An alternative would be to treat the single engine-type case the same as the multiple engine-types case: Use overlap documents to train multiple database-specific models for transforming database-specific document scores into database-independent document scores. More models would provide greater flexibility, but there would be less training data available to build each model, which might result in greater variance in model quality.

An experiment was conducted to test the hypothesis that combining the training data from each database to learn a single model would be superior to learning separate models for each database when all of the databases are searched with the same software. As in the previous experiment, all databases were searched by the INQUERY search engine. Table X summarizes the results.

The experiment supports the hypothesis that it is more effective to combine training data when possible. When the databases have similar vocabulary patterns, as on the trec123 testbed, training multiple database-specific models was clearly less effective than training a single model for all databases. When the databases were relatively dissimilar, as in the trec4_kmeans testbed, the two approaches were about equally effective. The results suggest that although multiple models have more tuning power than a single model, the reduced amount of training data counteracts the additional tuning power. When there is prior knowledge that the search engines are all of the same type, it is more effective to use all of the training data to build a single model.

## 7. EXPERIMENTAL RESULTS: THE EFFECT OF DIFFERENT RESOURCE SELECTION AND DOCUMENT RETRIEVAL ALGORITHMS

The experiments described above used the CORI resource selection algorithm to select which databases to search and the INQUERY document retrieval algorithm to find "overlap" information in the centralized sample database. These algorithms were chosen because they are very effective, and because the baseline CORI result-merging algorithm requires them. However, the SSL result-merging algorithm is not restricted to use with these algorithms. It can be used just as easily with other resource selection and document retrieval algorithms. This section investigates the flexibility and robustness of the SSL result-merging algorithm with a set of experiments that use language-modeling algorithms for resource selection and document retrieval.

The language-modeling approach to document retrieval was introduced briefly in Section 4.2 (Eq. (18)). The same algorithm was used to search the centralized sample database for "overlap" information in the experiments reported below.

The language-modeling approach can be adapted to database selection by treating each database as if it were a single big document [Si et al. 2002; Xu and Croft 1999]. Resource selection is modeled as selecting the databases that have the largest probabilities $P(C_i|Q)$. Bayes' Rule allows this probability to be represented as:

$$P(C_i|Q) = \frac{P(Q|C_i)P(C_i)}{P(Q)}, \tag{20}$$

where $P(Q|C_i)$ is the generation probability of the query Q given the language model[4] of the $i$th database $C_i$. P(Q) is the prior probability of observing the query; it is constant during database selection. We assume a uniform distribution on the collections, which makes $P(C_i)$ a constant, too. Hence, $P(C_i|Q)$ is proportional to $P(Q|C_i)$. $P(Q|C_i)$ is calculated as a mixture of two language models, as shown below:

$$P(Q|C_i) = \prod_{q \in Q} (\lambda P_{mle}(q|C_i) + (1-\lambda)P_{mle}(q|G)). \tag{21}$$

$P_{mle}(q|C_i)$ and $P_{mle}(q|G)$ are the maximum likelihood estimator (mle) language models for the $i$th database and the union of all databases [Si et al. 2002]. $\lambda$ is set to 0.5.

This approach to using language models for resource selection is similar to the Kullback–Leibler (KL) divergence-based collection selection method used by Xu and Croft [1999]. Prior research showed that Xu and Croft's KL divergence approach to database selection is about as effective as the CORI resource selection algorithm [Larkey et al. 2000; Xu and Croft 1999].

The effect of the language-modeling approaches to resource selection and document retrieval on the CORI and SSL result-merging algorithms was tested in experiments using the multiple search engine experimental methodology described above. Resource descriptions were created from the same sets of documents used in prior experiments with CORI resource selection and INQUERY document retrieval algorithms. Three types of search engines were used, ten databases were selected for each query and result-lists describing 1,000 documents were returned by each database. The constant in the CORI result-merging algorithm is not tuned for use with a KL-divergence resource selection algorithm, so a range of values was tested to provide a fair baseline. Figures 5 and 6 summarize the results.

[4]We treat *resource descriptions* and *language models* as equivalent concepts in this article. Both describe the vocabularies and term frequencies of terms in a sample of text. Usually resource descriptions describe term counts and language models describe probabilities of occurrence, but these are just different representations of the same information.

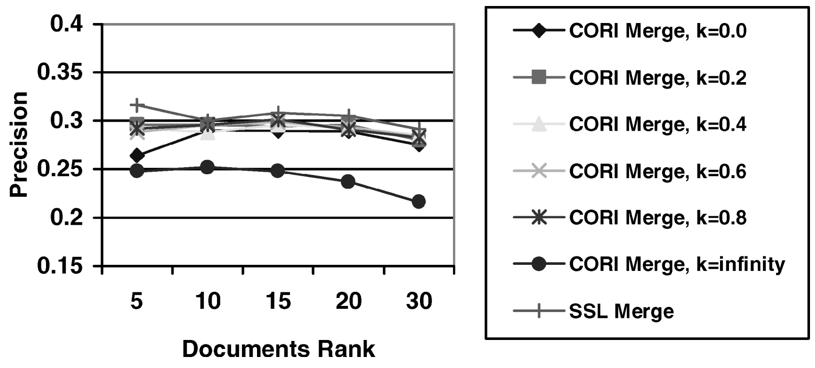

Fig. 5. How varying the *k* parameter in the CORI result-merging algorithm affects precision at different document ranks on the rec123 testbed. Language Model resource selection and Language Model central retrieval method were used for the semisupervised learning method.

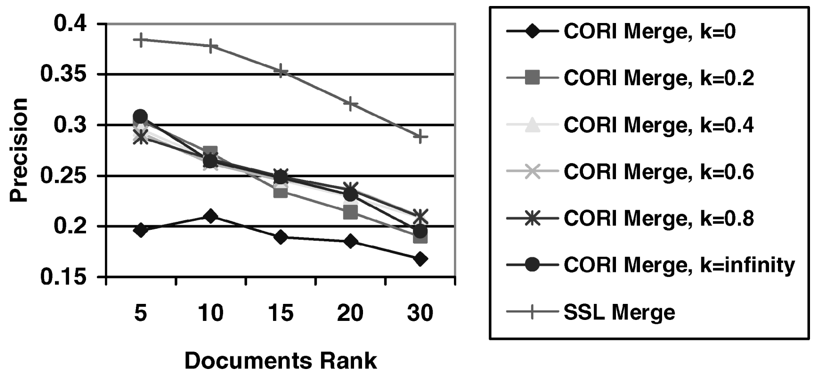

Fig. 6. How varying the *k* parameter in the CORI result-merging algorithm affects precision at different document ranks on the trec4-kmeans testbed. Language Model resource selection and Language Model central retrieval method were used for the semisupervised learning method.

The relative accuracy of the two result-merging algorithms in these experiments was similar to the results reported above, in Section 5.2. The SSL merging algorithm was slightly more effective than the CORI merge algorithm on the trec123 testbed, and substantially more effective than the CORI merge algorithm on the trec4_kmeans testbed.

These results are consistent with the conclusion that we drew in Section 5.2 that the SSL algorithm is always at least as good as the CORI merge, but tends to have a large advantage on the topic-organized testbed. This experiment confirms again that the power of the SSL algorithm lies in the query-by-query model tuning for different databases.

Careful comparison of Figures 5 and 6 with Figures 3 and 4 reveals that in these experiments the results for the trec4_kmeans testbed were much better than in the prior experiments. The difference was due in large part to more accurate resource selection, and is consistent with earlier research showing that the KL-divergence resource selection algorithm is more effective than CORI resource selection when databases are organized by topic [Larkey et al. 2000].

Table XI. Precision at Different Document Ranks using the CORI and Semisupervised Learning Approaches to Merging Retrieval Results. Resource Descriptions were Created from 300 Sampled Documents

| Document Rank | Trec123 Testbed | | Trec4_kmeans Testbed | |
|---|---|---|---|---|
| | CORI Merge | SSL Merge 300 sampled documents | CORI Merge | SSL Merge 300 sampled documents |
| 5 | 0.3280 | 0.3240 (−1.21%) | 0.2400 | 0.3360 (+40.00%) |
| 10 | 0.3020 | 0.3520 (+16.56%) | 0.1940 | 0.3120 (+60.82%) |
| 15 | 0.3107 | 0.3400 (+9.43%) | 0.1920 | 0.2720 (+41.67%) |
| 20 | 0.3000 | 0.3200 (+6.25%) | 0.1800 | 0.2580 (+43.33%) |
| 30 | 0.2873 | 0.2993 (+4.18%) | 0.1627 | 0.2300 (+41.36%) |

*Note*: Ten databases were selected to search for each query. Results are averaged over 50 queries.

The experiments reported in this section demonstrate that the SSL result-merging algorithm is not restricted to use with the CORI resource selection and INQUERY document retrieval algorithms. At least one other algorithm can be used for each of these tasks. It is likely that other resource selection and document retrieval algorithms can be used with the SSL result-merging algorithm, too.

## 8. EXPERIMENTAL RESULTS: QUERY-BASED SAMPLING EFFECTS

Prior research showed that the number of documents examined during query-based sampling has a direct, although diminishing, effect on resource selection accuracy [Callan 2000]. The number of documents examined during query-based sampling also determines the size of the centralized sample database that provides training data for the SSL result-merging algorithm. The single engine-type experiments described in Section 6 suggest that the accuracy of the SSL result-merging algorithm is affected in part by the amount of training data available for learning models. In this section, we examine the sensitivity of the SSL result-merging algorithm to the number of documents examined during query-based sampling.

A series of experiments was done to study the effects of varying the number of documents examined during query-based sampling. The experiments were designed to distinguish among the influences of the number of documents on resource selection and on result merging. The multiengine experimental methodology described in Section 5.2 was used: Three types of search engines, CORI resource selection, and ten databases selected per query. Two sample sizes were compared: Our default sample size of 300 documents per database and a larger size of 700 documents per database (obtained using about 175 queries). As in our prior experiments, a random sampling strategy was used to select query terms during query-based sampling, which means that two equal-sized samples will produce slightly different resource descriptions and slightly different results. In these experiments, the 300-document samples are the first 300 documents observed in a 700-document sampling run, which ensures that the 300-document and 700-document results are directly comparable. Tables XI and XII summarize the results.

The 300-document results (Table XI) were consistent with results reported earlier in this paper (Table VI). The numbers varied slightly, due to random

Table XII. Precision at Different Document Ranks using the CORI and Semisupervised Learning Approaches to Merging Retrieval Results. Resource Descriptions were Created from 700 Sampled Documents

| Document Rank | Trec123 Testbed | | Trec4_kmeans Testbed | |
|---|---|---|---|---|
| | CORI Merge | SSL Merge 700 sampled documents | CORI Merge | SSL Merge 700 sampled documents |
| 5 | 0.3280 | 0.3880 (+18.3%) | 0.2600 | 0.3800 (+46.2%) |
| 10 | 0.3400 | 0.3640 (+7.1%) | 0.2160 | 0.3320 (+53.7%) |
| 15 | 0.3360 | 0.3520 (+4.8%) | 0.1947 | 0.3107 (+59.6%) |
| 20 | 0.3260 | 0.3420 (+4.9%) | 0.1850 | 0.2880 (+55.7%) |
| 30 | 0.3100 | 0.3133 (+1.1%) | 0.1700 | 0.2587 (+52.5%) |

*Note*: Ten databases were selected to search for each query. Results are averaged over 50 queries.

term selection during query-based sampling, but the results were qualitatively similar: The SSL merge was slightly better than the CORI merge on the trec123 testbed and much better than the CORI merge on the trec4_kmeans testbed.

The results for both result-merging algorithms were better when 700 documents per database were sampled than when 300 documents per database were sampled. The only possible reason that the CORI merging results were better is that resource selection is more accurate when resource descriptions are created from 700 documents. The results for the SSL algorithm could be caused by improved resource selection and/or by improved merging due to a larger amount of training data (more "overlap" information). A second experiment was conducted to distinguish among these effects.

The second experiment was conducted only with the SSL method. In addition to the experimental configurations used in the previous experiment, two new configurations were introduced that allocated unequal amounts of training data to resource selection and result merging. One new configuration used 700 sampled documents per database for resource selection, but only 300 sampled documents per database were available to the SSL result-merging algorithm. The other new configuration used 300 documents per database for resource selection, but 700 documents per database were available to the SSL algorithm. The unequal allocations of sampled data make it possible to test the effects of increased data on just one component of the system. The results are summarized in Tables XIII and XV.

The more detailed experiment indicated that improved result merging was responsible for about half of the improvement observed in Tables XIII and XV when 700 documents were sampled per database. When resource descriptions were based on 300 documents per database and the centralized sample database was based on 700 documents per database (Tables XIII and XV, 3rd column of results), Precision increased about 4% for the trec123 testbed and about 9% for the trec4_kmeans testbed. This increase could only be due to improved result-merging due to a larger number of "overlap" documents.

The more detailed experiment also indicated that the SSL result-merging algorithm is sensitive to mismatches in the information used for resource selection and result-merging. When the information used for resource selection was more accurate than the information used for result-merging (Tables XIII and XV, 2nd column of results), Precision decreased slightly on both testbeds.

Table XIII. Precision at Different Document Ranks using the Semisupervised Learning Approach to Merging Retrieval Results

| | Trec123 Testbed | | | |
|---|---|---|---|---|
| Document Rank | SSL Merge, 300 docs for selection, 300 docs for merging | SSL Merge, 700 docs for selection, 300 docs for merging | SSL Merge, 300 docs for selection, 700 docs for merging | SSL Merge, 700 docs for selection, 700 docs for merging |
| 5 | 0.3240 | 0.3640 (+12.4%) | 0.3560 (+9.88%) | 0.3880 (+19.75%) |
| 10 | 0.3520 | 0.3400 (−3.5%) | 0.3600 (+2.27%) | 0.3640 (+3.41%) |
| 15 | 0.3400 | 0.3280 (−3.5%) | 0.3427 (+0.79%) | 0.3520 (+3.53%) |
| 20 | 0.3200 | 0.3140 (−1.9%) | 0.3300 (+3.13%) | 0.3420 (+6.88%) |
| 30 | 0.2993 | 0.3020 (+0.9%) | 0.3107 (+3.81%) | 0.3133 (+4.68%) |

*Note*: Resource descriptions were created from 300 or 700 sampled documents. The centralized sample database was created from 300 or 700 sampled documents from each database. 10 databases were selected to search for each query. Results are averaged over 50 queries.

Table XIV. The Number of "Overlap" Documents per Database on the Trec123 Testbed

| | Trec123 Testbed | | | |
|---|---|---|---|---|
| | 300 docs for selection, 300 docs for merging | 700 docs for selection, 300 docs for merging | 300 docs for selection, 700 docs for merging | 700 docs for selection, 700 docs for merging |
| Available | 22.6 | 20.8 | 33.6 | 34.6 |
| Used | 9.06 | 8.96 | 9.41 | 9.48 |

*Note*: At most 10 "overlap" documents per database are used for training, so the number used can be less than the number available. Ten databases are selected per query. Results are averaged over 50 queries.

Resource selection was more accurate in this case, so the loss could only be due to the SSL result-merging algorithm. Further analysis indicated that in this situation the resource selection algorithm was more likely to select (good) databases for which the result-merging algorithm was unable to obtain as much training data ("overlap" information) as other configurations. In these cases, weak result-merging offsets any improvement in resource selection. The number of training data ("overlap" documents) is shown in Tables XIV and XVI. The configuration of 700 documents for resource selection and 300 documents for result-merging (Tables XIV and XVI, 2nd column of results) had the least training data. Although the difference in the amount of *used* training data between this configuration and other configurations is not very distinct, there is a larger difference in the amount of *available* training data. Generally, more available training data enables the result-merging algorithm to choose better training data points (the training data with higher ranks), which helps to improve retrieval accuracy.

The experiments reported in this section indicate that increasing the number of documents sampled per database can improve result-merging accuracy, but is not guaranteed to do so. The most accurate results were obtained when resource descriptions and the centralized sample database were constructed from the

Table XV. Precision at Different Document Ranks using the Semisupervised Learning Approach to Merging Retrieval Results

| Document Rank | Trec4_Kmeans Testbed | | | |
|---|---|---|---|---|
| | SSL Merge, 300 docs for selection, 300 docs for merging | SSL Merge, 700 docs for selection, 300 docs for merging | SSL Merge, 300 docs for selection, 700 docs for merging | SSL Merge, 700 docs for selection, 700 docs for merging |
| 5 | 0.3360 | 0.3040 (−9.5%) | 0.3560 (+5.95%) | 0.3800 (+13.10%) |
| 10 | 0.3120 | 0.2960 (−5.1%) | 0.3280 (+5.13%) | 0.3320 (+6.41%) |
| 15 | 0.2720 | 0.2573 (−5.4%) | 0.3160 (+16.18%) | 0.3107 (+14.23%) |
| 20 | 0.2580 | 0.2510 (−2.7%) | 0.2840 (+10.08%) | 0.2880 (+11.63%) |
| 30 | 0.2300 | 0.2273 (−1.2%) | 0.2520 (+9.57%) | 0.2587 (+12.40%) |

*Note*: Resource descriptions were created from 300 or 700 sampled documents. The centralized sample database was created from 300 or 700 sampled documents from each database. 10 databases were selected to search for each query. Results are averaged over 50 queries.

Table XVI. The Number of "Overlap" Documents per Database on the Trec4_kmeans Testbed

| | Trec4_Kmeans Testbed | | | |
|---|---|---|---|---|
| | 300 docs for selection, 300 docs for merging | 700 docs for selection, 300 docs for merging | 300 docs for selection, 700 docs for merging | 700 docs for selection, 700 docs for merging |
| Available | 56.0 | 37.5 | 79.9 | 77.4 |
| Used | 9.93 | 9.01 | 9.96 | 9.93 |

*Note*: At most 10 "overlap" documents per database are used for training, so the number used can be less than the number available. Ten databases are selected per query. Results are averaged over 50 queries.

same set of sampled documents, because there was a higher likelihood of finding sufficient "overlap" information for all of the selected databases.

## 9. EXPERIMENTAL RESULTS: THE EFFECT OF THE RESULT LIST LENGTH

In the previous experiments ranked lists of 1,000 documents were retrieved from each selected database. By ranked lists we mean document identifiers, ranks, and scores; the document text is not downloaded until a person requests it. Ranked list information can be communicated efficiently across a computer network. If the string identifier and numeric score for a document can be communicated in 60 bytes, then a ranked list of 1,000 documents requires about 60,000 bytes. The number of bytes varies depending upon the result list format, but in general the communications costs are not excessive.

Often result list lengths cannot be controlled by the merging process. For example, Google and AltaVista initially return only the top 10 or 20 documents. Additional communication is required to obtain results farther down the ranked list. If a search engine returns 20 documents at a time, obtaining information about the top 1,000 documents would require 50 "client sends request and server responds" transactions. We would view this as excessive communication.

Table XVII. The Percentage of Selected Databases That Do Not have Enough Overlap Documents, for Result Lists of Different Lengths

| Result List Length | Percentage of Selected Databases With Fewer Than 3 Overlap Documents | |
|---|---|---|
| | Trec123 testbed | Trec4_kmeans Testbed |
| 50 | 62.0% | 21.0% |
| 100 | 31.8% | 8.0% |
| 200 | 12.6% | 1.6% |
| 500 | 6.2% | 0.4% |
| 1000 | 3.0% | 0.0% |

*Note*: Ten databases were selected per query. Results are averaged over 50 queries.

In many operational environments, it is desirable to rely on result-lists far shorter than 1,000 documents. However, one consequence of using shorter result-lists, for example, 50 documents, is that the semi-supervised learning approach to result merging is likely to have much less training data, that is, less "overlap" information, from which to learn its models. The multiengine version of the algorithm needs "overlap" information for at least three documents for each database. A very real concern is that the training data requirements of the SSL method can't be satisfied inexpensively in environments where search engines return only short result lists.

The effect of the result list length on result-merging accuracy was explored in a series of experiments. The multiple search engine types experimental methodology described in previous sections was used in these experiments, except that the length of result lists was varied from 50 to 1,000. All search engines were assumed to return lists of the same length. The results are summarized in Table XVII.

When the result list described only the top 50–100 documents from each database, a large percentage of the selected databases did not have enough overlap information to provide sufficient training data. The problem was less severe on the trec4_kmeans testbed, which has homogeneous databases with shorter average length and longer queries; result lists of 100–200 documents were sufficient to provide adequate training data for all but a small percentage of databases. The problem was more severe on the trec123 testbed, which has many somewhat similar heterogeneous databases; result lists needed to contain about 500 documents.

It is encouraging that in these experiments shorter result lists worked well with the trec4_kmeans testbed, which is usually considered the more difficult testbed for result-merging algorithms. One might conclude that when relevant documents are distributed across a small number of databases, short result lists are acceptable, but when relevant documents are scattered across a larger number of relatively similar databases, longer result lists are required. However, our goal is more consistent behavior from a result-merging algorithm.

Additional "overlap" information can be created on-the-fly by allowing the result-merging process to download a small number of documents from databases that do not have enough "overlap" information. The downloaded

Table XVIII. The Average Number of Downloaded Documents Required to Provide Overlap Information about 3 Documents per Selected Database

| | Trec123 Testbed | | Trec4_kmeans Testbed | |
|---|---|---|---|---|
| Result List Length | Download Docs Per Database | Total Download Docs (10 Databases) | Download Docs Per Database | Total Download Docs (10 Databases) |
| 50 | 1.2 | 12.1 | 0.4 | 3.8 |
| 100 | 0.5 | 5.3 | 0.1 | 1.3 |

*Note*: Ten databases were selected per query. Results are averaged over 50 queries.

documents can be inserted into the centralized sample database (temporarily or permanently), and scores can be computed for them. This solution has communications costs (i.e., the cost of downloading the additional documents), but the cost of downloading a few documents may be much lower than the cost of obtaining the many result-list fragments needed to reach 200–500 documents per database.

The experiment described above was repeated, except that if overlap information was not available for at least three documents in a result list, enough documents were downloaded from the database to provide overlap information for three of its documents. Documents ranked 1st, 11th, and 21st were candidates for downloading. These ranks were chosen to cover a small range of top-ranked document scores. Documents that were downloaded on-the-fly were inserted into the centralized sample database temporarily so that scores could be computed for them.[5] The experimental results are summarized in Table XVIII.

When result lists contained information about the 50 top ranked documents, the minimum training data requirements were met by downloading an average of 0.4–1.2 documents per database. When result lists contained information about the 100 top-ranked documents, it was only necessary to download an average of 0.1–0.5 documents per database. In these experiments, the costs of downloading documents to meet the minimum training data requirements were relatively low, and almost certainly lower than the cost of downloading additional fragments of the result lists.

The effectiveness of this approach to meeting the minimum training data requirements is shown in Table XIX. The combination of shorter result lists and on-the-fly document downloading was about as effective as result lists of 1,000 documents. The variance in effectiveness was greater when result lists were short and documents were downloaded on-the-fly; the increased variance was due to the reduced amount of training data. However, in general the difference was too small for an interactive user to notice. Effectiveness was maintained, in spite of greatly reduced training data because the algorithm could choose which documents to download. It could focus its limited amount of training on the higher-ranked documents, which was a more efficient use of training data

[5]Documents were inserted into the centralized sample database only temporarily during these experiments to prevent one query from affecting subsequent queries. We return to this point in Section 10.

Table XIX. Precision at Different Document Ranks for Three Methods of Supplying Training Data to the Semisupervised Learning Algorithm for Merging Retrieval Results

| Document Rank | Trec123 Testbed | | | Trec4_kmeans Testbed | | |
|---|---|---|---|---|---|---|
| | 1,000 Docs | Top 50 Docs +Downloads | Top 100 Docs +Downloads | 1,000 Docs | Top 50 Docs +Downloads | Top 100 Docs +Downloads |
| 5 | 0.3520 | 0.3680 (+4.45%) | 0.3440 (−2.27%) | 0.3640 | 0.3480 (−4.40%) | 0.3520 (−3.30%) |
| 10 | 0.3400 | 0.3600 (+5.88%) | 0.3460 (+1.76%) | 0.2940 | 0.2980 (+1.36%) | 0.2960 (+0.68%) |
| 15 | 0.3280 | 0.3587 (+9.36%) | 0.3360 (+2.44%) | 0.2760 | 0.2667 (−3.37%) | 0.2627 (−4.82%) |
| 20 | 0.3290 | 0.3500 (+6.38%) | 0.3340 (+1.52%) | 0.2540 | 0.2530 (−0.39%) | 0.2520 (−0.79%) |
| 30 | 0.3200 | 0.3340 (+4.38%) | 0.3160 (−1.25%) | 0.2300 | 0.2233 (−2.91%) | 0.2240 (−2.61%) |

*Note*: Each method is characterized by a result-list length (1000, 50, and 100) and whether documents can be downloaded on-the-fly to generate additional "overlap" information. Ten databases were selected per query. Results are averaged over 50 queries.

Table XX. The Number of "Overlap" Documents per Database

| Result List Length | Overlap Documents Per Database | | | |
|---|---|---|---|---|
| | Trec123 Testbed | | Trec4_kmeans Testbed | |
| | Available | Used | Available | Used |
| Top 50 Docs + Downloads | 3.9 | 3.8 | 5.7 | 5.5 |
| Top 100 Docs + Downloads | 5.3 | 4.6 | 10.3 | 8.0 |
| 1,000 Docs | 24.5 | 9.1 | 57.8 | 9.9 |

*Note*: At most 10 "overlap" documents per database are used for training, so the number used can be less than the number available. Ten databases are selected per query. Results are averaged over 50 queries.

(Table XX), and which maintained accuracy in the portions of the rankings most likely to be observed by the user.

These experiments demonstrate that the semisupervised learning approach to merging results can be applied in environments where result lists are short and communications costs are to be kept low. Allowing the algorithm to download a very small number of documents per database compensates for dramatically shorter result lists with no significant effect on accuracy.

## 10. CONCLUSIONS

This article presents the results of an extended study of a semisupervised learning (SSL) approach to result-list merging for autonomous, uncooperative search engines in distributed information retrieval environments. The SSL algorithm assumes that result merging is an integrated part of a complete distributed information retrieval system, and that information acquired for creating resource descriptions can also be used to guide result merging. The SSL algorithm specifically models result merging as a task of transforming sets of database-specific document scores into a single set of database-independent document scores. We call this method a *semisupervised learning* approach because data acquired for creating resource descriptions is automatically used by an ordinary document retrieval algorithm to create training data that teaches a supervised learning algorithm to normalize the scores of unseen documents. We believe that this is the first result-merging algorithm to model the task in this way and to learn database-specific and query-specific functions for normalizing document scores.

The research reported here reproduces results reported in an earlier paper [Si and Callan 2002]. In particular, it demonstrates that the algorithm is effective on two rather different 100-database testbeds under a variety of conditions, in environments with a single type of search engine, and in environments with several types of search engines. The SSL algorithm is shown to be at least as effective as the well-known CORI result-merging algorithm, and its effectiveness is shown to be due to its ability to tune a linear normalization model on a database-specific and query-specific basis.

This article also extends the prior research in several important directions. It addresses the criticism that the SSL result-merging algorithm might be impractical in operational environments because it relies on receiving result-list information about a large number of documents (1,000) from each database. The research reported here demonstrates that the algorithm is effective with relatively short result-lists if it is also allowed to download a small number of document texts "on the fly" from selected databases. In this configuration the algorithm uses its training data very efficiently, which gives implementers considerable freedom in how to trade off result-list length and the number of documents downloaded "on the fly."

The earlier paper [Si and Callan 2002] claimed that the SSL result-merging framework is independent of the particular resource selection algorithm and the algorithm used to search the centralized sample database. The research reported here supports that claim by demonstrating effective results with language-modeling algorithms for resource selection and document retrieval from the centralized sample database. We believe that the SSL result-merging framework can be used with any reasonably effective resource selection and document retrieval algorithms.

The SSL result-merging algorithm increases the importance of query-based sampling, because the sampled documents are used for two purposes: building the resource descriptions that guide resource selection, and building the centralized sample database that supplies training data for learning to merge result lists. This article shows that the effect of varying the amount of sampled data is magnified because of this dual use. Increasing the number of sampled documents improves resource selection accuracy, and also improves the accuracy of result merging.

The SSL algorithm is also sensitive to inconsistencies between the amount of information used for resource selection and merging results, which was not known previously. If the resource selection algorithm makes its decisions based on more or different information than is used by the SSL algorithm, any improvement in resource selection accuracy may be lost due to an inability to find comparable training data for result-merging in the centralized sample database. This problem can probably be solved by allowing the SSL algorithm to dynamically download document "on the fly," as is done when result lists are short, but that solution remains to be studied in future work.

As with any new algorithm, there remain open questions about this approach to merging results. For example, we have studied just one type of method for learning to normalize document scores. Linear models are a good choice when the training data are limited, but occasionally, for some combinations of queries and databases, training data is plentiful. Perhaps more complex models can be adopted when there is more training data. In our research, the centralized sample database did not change over time, but in an operational environment the centralized sample database probably would change over time, for example, each time it is necessary to download documents "on the fly" to generate additional training data. Over time, the centralized sample database might be expected to become a very good model of the information most relevant to the queries that are seen most often, which would enable, for example, a gradual transition from simple models to complex models. We are not aware of any research on distributed information retrieval systems that learn over time, but this seems likely to be a fruitful area for future research.

The result-merging problem in distributed information retrieval environments, especially in environments with multiple types of uncooperative search engines, has been a major open problem for several years. Although it is not yet completely solved, we view the work presented in this article as an important improvement in the state-of-the-art.

## REFERENCES

Aslam, J. A. and Montague, M. 2001. Models for metasearch. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York.

Buckley, C., Singhal, A., Mitra, M., and Salton, G. 1995. New retrieval approaches using SMART. In *Proceedings of 1995 Text REtrieval Conference (TREC-3)*, special publication. National Institute of Standards and Technology.

Callan, J. 2000. Distributed information retrieval. In *Advances in Information Retrieval*, W. B. Croft, Ed. Kluwer Academic Publishers, pp. 127–150.

Callan, J. and Connell, M. 2001. Query-based sampling of text databases. *ACM Trans. Inf. Syst. 19*, 2, 97–130.

Callan, J., Croft, W. B., and Broglio, J. 1995a. TREC and TIPSTER experiments with INQUERY. *Inf. Proc. Manage. 31*, 3, 327–343.

Callan, J., Lu, Z., and Croft, W. B. 1995b. Searching distributed collections with inference networks. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York.

Craswell, N., Bailey, P., and Hawking, D. 2000. Server selection on the World Wide Web. In *Proceedings of the 5th ACM Conference on Digital Libraries*. ACM, New York, pp. 37–46.

Craswell, N., Hawking, D., and Thistlewaite, P. 1999. Merging results from isolated search engines. In *Proceedings of the 10th Australasian Database Conference*. pp. 189–200.

French, J. C., Powell, A. L., Callan, J., Viles, C. L., Emmitt, T., Prey, K. J., and Mou, Y. 1999. Comparing the performance of database selection algorithms. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York.

Fuhr, N. 1999. A decision-theoretic approach to database selection in networked IR. *ACM Trans. Inf. Syst. 17*, 3, 229–249.

Gravano, L., Chang, C., Garcia-Molina, H., and Paepcke, A. 1997. STARTS: Stanford proposal for Internet Meta-Searching. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data.* ACM, New York.

Gravano, L., Garcia-Molina, H., and Tomasic, A. 1994. The effectiveness of GlOSS for the text database discovery problem. In *Proceedings of the ACM-SIGMOD International Conference on Management of Data.* ACM, New York.

Gravano, L., Garcia-Molina, H., and Tomasic, A. 1999. GlOSS: Text-Source discovery over the Internet. *ACM Trans. Datab. Syst. 24*, 2.

HAWKING, D. AND THISTLEWAITE, P. 1999. Methods for information server selection. *ACM Trans. Inf. Syst. 17*, 1, 40–76.

IPEIROTIS, P. AND GRAVANO, L. 2002. Distributed search over the Hidden-Web: Hierarchical database sampling and selection. In *Proceedings of the 28th International Conference on Very Large Databases (VLDB).*

KIRSCH, S. T. 2003. Document retrieval over networks wherein ranking and relevance scores are computed at the client for multiple database documents. U.S. Patent 5,659,732.

LARKEY, L., CONNELL, M., AND CALLAN, J. 2000. Collection selection and results merging with topically organized U.S. patents and TREC data. In *Proceedings of Conference of Information and Knowledge Management.*

LE CALV, A. AND SAVOY, J. 2000. Database merging strategy based on logistic regression. *Inf. Proc. Manage. 36*, 3.

LEE, J. H. 1997. Analyses of multiple evidence combination. In *Proceedings. of the 20th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, New York.

LEMUR TOOLKIT. 2003. http://www.cs.cmu.edu/∼lemur.

MANMATHA, R., RATH, T., AND FENG, F. 2001. Modeling score distributions for combining the outputs of search engines. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, New York.

OGILVIE, P. AND CALLAN, J. 2001. Experiments using the Lemur toolkit. In*Proceedings of 2001 Text REtrieval Conference (TREC 2001)*, special publication. National Institute of Standards and Technology, Washington, DC.

PONTE, J. AND CROFT, W. B. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, New York.

POWELL, A. L., FRENCH, J. C., CALLAN, J., CONNELL, M., AND VILES, C. L. 2000. The impact of database selection on distributed searching. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, New York.

ROBERTSON, S. E. AND WALKER, S. 1994. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, New York, pp. 232–241.

SI, L. AND CALLAN, J. 2002. Using sampled data and regression to merge search engine results. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, New York.

SI, L., JIN, R., CALLAN, J., AND OGILVIE, P. 2002. A language model framework for resource selection and results merging. In *Proceedings of the 11th International Conference on Information and Knowledge Management (CIKM).* ACM, New York.

SONG, F. AND CROFT, W. B. 1999. A general language model information retrieval. In *Proceedings of the 22nd ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York.

VILES, C. L. AND FRENCH, J. C. 1995. Dissemination of collection wide information in a distributed information retrieval system. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York.

VOORHEES, E., GUPTA, N. K., AND JOHNSON-LAIRD, B. 1995. Learning collection fusion strategies. In *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, New York.

XU, J. AND CALLAN, J. 1998. Effective retrieval with distributed collections. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, New York.

XU, J. AND CROFT, W. B. 1999. Cluster-based language models for distributed retrieval. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval.* ACM, New York.

Received October 2002; revised January 2003; accepted July 2003